

CENTRO UNIVERSITÁRIO ASSIS GURGACZ

SANDRO MATHEUS RAMOS

TIPOS DE TESTES

DE SOFTWARE

CASCAVEL – PR

2021



CENTRO UNIVERSITÁRIO ASSIS GURGACZ

SANDRO MATHEUS RAMOS

TIPOS DE TESTES

DE SOFTWARE

Trabalho apresentado na Disciplina de Projeto de Programação Web, do Curso de Engenharia de Software do Centro Universitário Assis Gurgacz.

Professor: Elaine Zanini

CASCAVEL – PR

2021

[**1. INTRODUÇÃO** 5](#_Toc80521694)

[**2. O QUE É O SWEBOK?** 6](#_Toc80521695)

[**3. O QUE É TESTE DE SOFTWARE?** 7](#_Toc80521696)

[**4. FUNDAMENTOS DO TESTE DE SOFTWARE** 7](#_Toc80521697)

[**5. TIPOS DE TESTE DE SOFTWARE** 8](#_Toc80521698)

[5.1 TESTE DE UNIDADE (UNITÁRIOS) 8](#_Toc80521699)

[5.2 TESTE DE INTEGRAÇÃO 8](#_Toc80521700)

[5.3 TESTE DE SISTEMA 9](#_Toc80521701)

[5.4 TESTE DE CAIXA BRANCA 9](#_Toc80521702)

[5.5 TESTE DE CAIXA PRETA 9](#_Toc80521703)

[5.6 TESTE DE CAIXA CINZA 9](#_Toc80521704)

[5.7 TESTE DE REGRESSÃO 9](#_Toc80521705)

[5.8 TESTE DE CARGA 10](#_Toc80521706)

[5.8 TESTE DE USABILIDADE 10](#_Toc80521707)

[5.8 TESTE DE STRESS 10](#_Toc80521708)

[**6. TÉCNICAS DE TESTE DE SOFTWARE** 10](#_Toc80521709)

[6.1 BASEADO NA INTUIÇÃO E EXPERIÊNCIA DO ENGENHEIRO DE SOFTWARE 10](#_Toc80521710)

[6.1.1 Ad Hoc 11](#_Toc80521711)

[6.1.2 Teste exploratório 11](#_Toc80521712)

[6.2 BASEADA NA ESPECIFICAÇÃO TÉCNICA 11](#_Toc80521713)

[6.2.1 Equivalência de particionamento 11](#_Toc80521714)

[6.2.2 Análise de valores limites 11](#_Toc80521715)

[6.2.3 Tabela de decisão 12](#_Toc80521716)

[6.2.4 Especificações formais 12](#_Toc80521717)

[6.2.5 Testes aleatórios 12](#_Toc80521718)

[6.3 BASEADO NO CÓDIGO 12](#_Toc80521719)

[6.3.1 Baseados em controle de fluxo 12](#_Toc80521720)

[6.3.2 Baseado no fluxo de dados 13](#_Toc80521721)

[6.3.3 Referência para modelos 13](#_Toc80521722)

[6.4 BASEADA EM FALHAS 13](#_Toc80521723)

[6.4.1 Adivinhar erros 13](#_Toc80521724)

[6.4.2 Teste de mutação 13](#_Toc80521725)

[6.5 BASEADA NO USO 14](#_Toc80521726)

[6.5.1 Perfil operacional 14](#_Toc80521727)

[6.5.2 Heurística 14](#_Toc80521728)

[6.6 BASEADA NA NATUREZA DA APLICAÇÃO 14](#_Toc80521729)

[6.7 SELEÇÃO E COMBINAÇÃO 14](#_Toc80521730)

[6.7.1 Funcional e estrutural 14](#_Toc80521731)

[6.7.2 Determinística e aleatória 15](#_Toc80521732)

[**7. CONCLUSÃO** 15](#_Toc80521733)

[**8. REFERÊNCIAS** 15](#_Toc80521734)

# **1. INTRODUÇÃO**

Nas últimas décadas, a Engenharia de Software se subdividiu em diversas áreas, a quantidade de informação aumentou tanto que a constante especialização profissional se tornou algo comum para atingir o nível de excelência requerido pelo mercado, e profissionais que não se adaptam a essa constante evolução se tornam obsoletos em um curto período de tempo. O termo Engenharia de Software foi usado pela primeira como tema na conferência da OTAN no ano de 1968, evento esse que foi motivado pela crise no desenvolvimento de software. Nesta época, viu-se necessária a busca por métodos padronizados de desenvolvimento de software, visto que a grande maioria era construído sem padronização nenhuma. John Ronald Graham comentou: "construímos sistemas como os irmãos Wright construíam aviões - constrói-se de uma só vez, empurra-se para o despenhadeiro, deixa bater e começa tudo outra vez" (Naur & Randell, 1968).

No auxílio a esses profissionais e para atingir uma padronização foi criado o SWEBOK (Software Engineering Body of Knowledge), assim como o PMBOK(Project Management Body of Knowledge) ajuda os gerentes de projeto nas mais diversas áreas/disciplinas da gerencia de um projeto, o SWEBOK ajuda os Engenheiros de Software nas mais diversas áreas da construção de um software, ele é um guia para o corpo de conhecimento em Engenharia de Software, patrocinado pelo IEEE (Institute of Electrical and Electronics Engineering). Se tratando um artigo relacionado a testes de software, veremos a visão do SWEBOK quanto a testes, suas definições e de como atingir a qualidade provida pelos testes na entrega de um software (SWEBOK, 2004).

A atividade de teste não é uma tarefa simples, ela exige um bom planejamento durante a execução para ser bem sucedida. E a falta de conhecimento dos programadores ou analistas sobre o processo de teste de software pode fazer com que muitos erros passem despercebidos, resultando em um software sem qualidade, e assim trazendo insatisfação aos clientes (ONEDAYTESTING, 2017).

# **2. O QUE É O SWEBOK?**

Não podemos falar de uma disciplina(testes) de Engenharia de Software sem citar o SWEBOK, pois nele estão compilados vários anos de estudos e pesquisa em busca melhores práticas nas diversas disciplinas que ele aborda. Podemos dizer que ele é um guia completo com todo conhecimento atual sobre a Engenharia de Software, ele é organizado de modo sistemático e estruturado, esses conhecimentos são divididos em quinze áreas de conhecimentos, que são elas conforme SWEBOK (2004):

• Requisitos de Software;

• Design de Software;

• Construção de Software;

• Teste de Software;

• Manutenção de Software;

• Gerenciamento de Configuração de Software;

• Gerenciamento de Engenharia de Software;

• Processos de Engenharia de Software;

• Modelos e Métodos da Engenharia de Software;

• Qualidade de Software;

• Práticas Profissionais em Engenharia de Software;

• Economia em Engenharia de Software;

• Fundamentos de Computação;

• Fundamentos Matemáticos;

• Fundamentos de Engenharia;

Como podemos ver Teste de Software é uma das disciplinas que são englobadas pelo SWEBOK. Nos próximos tópicos abordaremos os temas que contemplam esta disciplina a fim de termos um melhor entendimento e uma visão geral sobre testes, seus níveis, tipos e como aplicá-los da forma correta.

# **3. O QUE É TESTE DE SOFTWARE?**

Como estamos dando enfase nas ideias propostas no SWEBOK veremos primeiramente sua definição sobre o que são testes.

Segundo o SWEBOK, Teste de Software “é uma disciplina de engenharia de software que trata da avaliação da qualidade do produto e da identificação de defeitos e problemas. Consiste na verificação do comportamento de um programa em um conjunto finito de casos de teste”. Também não podemos deixar de citar Pressman (1995, p. 724) outro grande autor que define testes como “conformidade a requisitos funcionais e de desempenho explicitamente declarado, a padrões de desenvolvimento claramente documentados e a características implícitas que são esperadas de todo software profissionalmente desenvolvido”.

Também temos de visualizar o software como elemento de sistema e os “custos” envolvidos associados às falhas de software são forças propulsoras para uma atividade de teste cuidadosa e bem planejada (PRESSMAN, 1995).

# **4. FUNDAMENTOS DO TESTE DE SOFTWARE**

Neste tópico do SWEBOK ele nos apresenta os conceitos e terminologias que formam a base de conhecimento para compreensão do papel da manutenção de software, como por exemplo, a natureza, categorias, custos, evolução de software, entre outras (SWEBOK, 2004).

Quanto a terminologia, podemos verificar que esta é utilizada de acordo com a norma IEEE Standard 610.12-1990, que nos apresenta 3 termos relacionados a teste de software, que são elas:

• Falha (do inglês fault ): causa do mal funcionamento

• Defeito (do inglês failure ): resultado indesejado

• Erro (do inglês error ): uma ação humana que produz um resultado incorreto.

# **5. TIPOS DE TESTE DE SOFTWARE**

Os diversos tipos de Teste de Software devem sempre ser executados durante o desenvolvimento do software e em sua manutenção. Cada um com seu aspecto e trazendo qualidade tanto para o nível micro quanto para o nível macro do software (SWEBOK, 2004).

## 5.1 TESTE DE UNIDADE (UNITÁRIOS)

Segundo o (SWEBOOK, 2004) os testes de unidade são aplicados a módulos ou funções e verificam o funcionamento de componentes isolados. Também na visão do site (ONEDAYTESTING, 2017) esse tipo de teste trata de testar as menores unidades de um software, de modo isolado, visando que todas funcionem corretamente.

## 5.2 TESTE DE INTEGRAÇÃO

Testa um grupo de módulos e verifica a integração entre componentes (SWEBOOK, 2004). Após realizar os testes de unidade, o próximo passo é verificar se tudo funciona de maneira integrada. Podendo até mesmo apresentarem incompatibilidades ao serem testadas em conjunto, mesmo após passarem nos testes de unidade (ONEDAYTESTING, 2017).

## 5.3 TESTE DE SISTEMA

Testa o sistema como um todo, geralmente sob o ponto de vista do usuário, normalmente testando requisitos não funcionais (segurança, desempenho, integridade, etc)(SWEBOOK, 2004).

## 5.4 TESTE DE CAIXA BRANCA

Utiliza o aspecto interno do sistema como o código fonte, para avaliar seus componentes. Também conhecido como teste orientado a lógica estrutural. Analisados itens como: fluxo de dados, condições, loops. Ao implementá-lo é necessário verificar a criticidade, complexidade, a estrutura e nível de qualidade que se pretende obter do software (ONEDAYTESTING, 2017).

## 5.5 TESTE DE CAIXA PRETA

Teste de Caixa Preta que diferentemente do anterior, se baseia em aspectos externos, avaliando por exemplo os requisitos funcionais. Não leva em conta o modo de funcionamento e sim foca nas funções que deverão ser desempenhadas pelo software. Verificando se um grupo de entradas, resultou na saída esperada (ONEDAYTESTING, 2017).

## 5.6 TESTE DE CAIXA CINZA

Testes de Caixa Cinza, basicamente são a junção dos testes de Caixa Branca e Caixa Preta (ONEDAYTESTING, 2017).

## 5.7 TESTE DE REGRESSÃO

Teste de regressão, consiste em realizar testes a cada nova versão do sistema, de modo que erros antigos voltem a aparecer no software (ONEDAYTESTING, 2017).

## 5.8 TESTE DE CARGA

Teste de Carga, realizado para verificar qual o limite do software, quanto suporta de tráfego de informações, usuários logados e etc (ONEDAYTESTING, 2017).

## 5.8 TESTE DE USABILIDADE

Teste de Usabilidade, realizado por um grupo de usuários finais do sistema, de modo a verificar se o sistema os atende e colher feedback para futuras melhorias (ONEDAYTESTING, 2017).

## 5.8 TESTE DE STRESS

Teste de Stress, este leva o software ao seu limite de funcionamento e de potência, para assim avaliar em que ponto do uso ele deixaria de funcionar corretamente, geralmente realizado para verificar se as especificações mínimas e máximas são as adequadas (ONEDAYTESTING, 2017).

# **6. TÉCNICAS DE TESTE DE SOFTWARE**

Existem muitas definições para classificar todas as técnicas de software, mas as mais utilizadas e apresentadas no SWEBOOK serão abordadas nos próximos tópicos.

## 6.1 BASEADO NA INTUIÇÃO E EXPERIÊNCIA DO ENGENHEIRO DE SOFTWARE

### 6.1.1 Ad Hoc

O Ad Hoc é uma técnica derivada da habilidade, intuição e habilidade do engenheiro de software em programas semelhantes, muito útil para encontrar casos de teste que não são facilmente encontrados por técnicas mais formais (SWEBOOK, 2004).

### 6.1.2 Teste exploratório

O teste exploratório é definido como um aprendizado simultâneo, onde os testes e casos de teste não são criados com antecedência e os testes são projetado, executados e modificados dinamicamente (SWEBOOK, 2004).

## 6.2 BASEADA NA ESPECIFICAÇÃO TÉCNICA

### 6.2.1 Equivalência de particionamento

Baseia-se na em particionar o domínio de entrada em uma coleção de subconjuntos com base em um critério. Esse critério pode ser resultado computacionais diferentes, uma relação baseada no fluxo de controle ou de dados (SWEBOOK, 2004).

### 6.2.2 Análise de valores limites

Testes aonde são escolhidos valores limites do domínio de entrada das variáveis, com o raciocínio de que muitas falhas tendem a se concentrar perto dos valores de extremos de entradas (SWEBOOK, 2004).

### 6.2.3 Tabela de decisão

São sistematicamente derivados da criação de tabelas de decisão lógicas entre condições de entrada e ações de saída, gerando algo como causa e efeito, de uma forma lógica (SWEBOOK, 2004).

### 6.2.4 Especificações formais

A descrição formal dos casos de teste que assim permita a geração automática de casos de teste funcionais e também uma base para a verificação do resultado do mesmo (SWEBOOK, 2004).

### 6.2.5 Testes aleatórios

Baseia-se de uma forma simples na inserção de entradas aleatórias no domínio de entrada, de tal modo a gerar os mais diversos resultados no software, alguns desses até descobrindo falhas de segurança (SWEBOOK, 2004).

## 6.3 BASEADO NO CÓDIGO

### 6.3.1 Baseados em controle de fluxo

Visam cobrir todas as instruções, blocos de instruções ou combinações especificas de instruções de um software. O critério mais conhecido do fluxo de controle é o teste de caminho, onde executamos todos os caminhos possíveis daquela determinada instrução (SWEBOOK, 2004).

### 6.3.2 Baseado no fluxo de dados

Esse tipo de técnica o fluxo de controle é anotado com informações sobre como as variáveis do programa são definidas, usadas e eliminadas. O principal critério é o de que todos os caminhos de uso de definição, exigindo que, para cada variável, todo o cominho do fluxo de controle, da definição até seu uso, seja executado (SWEBOOK, 2004).

### 6.3.3 Referência para modelos

Mesmo não sendo uma técnica em si, a estrutura de controle de um programa pode ser representada graficamente usando um gráfico de fluxo para visualizar técnicas de teste baseadas em código (SWEBOOK, 2004).

## 6.4 BASEADA EM FALHAS

### 6.4.1 Adivinhar erros

Nesta técnica os casos de testes não são desenvolvidos por engenheiros de software que tentam antecipar os possíveis erros do sistema. Geralmente realizada utilizando o histórico de erros de projetos anteriores e a experiência do engenheiro de software (SWEBOOK, 2004).

### 6.4.2 Teste de mutação

Um mutante é uma versão ligeiramente modificada do software, se diferindo por uma pequena alteração sintática. Cada caso de teste executa funções do programa original, e os mutantes gerados, se um caso de teste for bem sucedido na identificação da diferença entre o software original e o mutante, o último é considerado “morto”, O efeito de acoplamento, é que, procurando por falhas sintáticas simples, falhas mais complexas e reais, serão encontradas (SWEBOOK, 2004).

## 6.5 BASEADA NO USO

### 6.5.1 Perfil operacional

Essa técnica se baseia na máxima aproximação do ambiente de testes ao ambiente operacional do software ou o perfil operacional. O objetivo é verificar os resultados dos testes quanto a confiabilidade futura do software quando em uso real (SWEBOOK, 2004).

### 6.5.2 Heurística

Os princípios de usabilidade podem fornecer diretrizes para a detecção de problemas na interface do usuário. As heurísticas especializas, são conhecidos métodos de verificação da usabilidade, são utilizadas na observação sistemática do uso do sistema sob condições controladas, a fim de verificar o quão simples e intuitivo a utilização das interfaces do software (SWEBOOK, 2004).

## 6.6 BASEADA NA NATUREZA DA APLICAÇÃO

Todas as técnicas já abordadas são aplicadas a todos os tipos de software. Técnicas adicionais são derivadas da natureza do software que está sendo testado, como por exemplo Software embarcado, Software orientado a objetos e Software baseado na Web.

## 6.7 SELEÇÃO E COMBINAÇÃO

### 6.7.1 Funcional e estrutural

Não são técnicas em si, mas sim complementos, que aproveitando da junção de duas técnicas, que destacam dois tipos de problemas diferentes. Utiliza a combinação da técnica de teste baseada no modelo funcional e estrutural e baseadas em código, frequentemente citadas como testes funcionais versus testes estruturais (SWEBOOK, 2004).

### 6.7.2 Determinística e aleatória

A técnica utilizada nos casos de testes pode ser selecionada de maneira determinística, de acordo com uma das muitas técnicas, ou aleatoriamente, provenientes de alguma distribuição de inputs, como é normalmente feito em testes de confiabilidade (SWEBOOK, 2004).

# **7. CONCLUSÃO**

Verificamos que os autores abordados ressaltam a importância de se verificar se o software apresenta suas as características esperadas, e o modo de sanar tal dúvida é com um bom conjunto de testes. Que para desenvolvermos software de qualidade que agrade as todas as partes interessadas, nós como “Engenheiros de Software” temos de ter conhecimento quanto a testes, e em nosso auxilio temos por exemplo o o SWEBOK, que nos traz uma série de tipos de testes, para entregarmos qualidade do código fonte até a usabilidade de nossas interfaces. Porém não basta somente ter o conhecimento dos tipos de teste existentes, também precisamos saber aplicá-los, nisso um SWEBOOK também dispõe de uma série de técnicas para a aplicação dos casos de testes.

# **8. REFERÊNCIAS**

IEEE, **IEEE Standard Glossary of Software Engineering Terminology – IEEE Std 610.12.1990**, 1990, IEEE, New York, EUA.

SWEBOK 2004, **Guide for the Software Engineering Body of Knowledge**, 2004 version, IEEE Computer Society, California, EUA.

PRESSMAN, R.S. **Engenharia de software**. São Paulo: Makron Books. 1995.

NAUR, RANDELL. **Software Engineering: Report on a Conference Sponsored by the NATO Science Committee. Technical report**, NATO, Garmisch, Germany.

ONE DAY TESTING. **TESTE DE SOFTWARE: INTRODUÇÃO, CONCEITOS BÁSICOS E TIPOS DE TESTES**. 2017. Disponível em: <https://blog.onedaytesting.com.br/teste-de-software/>. Acesso em: 20 agosto 2021.